

Package ‘SPAS’

October 12, 2022

Type Package

Title Stratified-Petersen Analysis System

Version 2020.1.1

Date 2020-01-01

Author Carl James Schwarz

Maintainer Carl James Schwarz <cschwarz.stat.sfu.ca@gmail.com>

LinkingTo TMB, RcppEigen

Imports BB, MASS, Matrix, msm, numDeriv, plyr, TMB (>= 1.7.15)

Description The Stratified-Petersen Analysis System (SPAS) is designed to estimate abundance in two-sample capture-recapture experiments where the capture and recaptures are stratified. This is a generalization of the simple Lincoln-Petersen estimator.

Strata may be defined in time or in space or both, and the s strata in which marking takes place may differ from the t strata in which recoveries take place.

When $s=t$, SPAS reduces to the method described by Darroch (1961) <<https://www.jstor.org/stable/2332748>>.

When $s<t$, SPAS implements the methods described in Plante, Rivest, and Tremblay (1988)

<<https://www.jstor.org/stable/2533994>>.

Schwarz and Taylor (1998) <[doi:10.1139/f97-238](https://doi.org/10.1139/f97-238)> describe the use of SPAS in estimating return of salmon stratified by time and geography.

A related package, BTSPAS, deals with temporal stratification where a spline is used to model the distribution of the population over time as it passes the second capture location.

This is the R-version of the (now obsolete) standalone Windows program available at <http://www.cs.umanitoba.ca/~popan/spas/spas_home.html>.

License GPL (>= 2)

RoxygenNote 7.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-12-02 09:30:11 UTC

R topics documented:

SPAS.fit.model	2
SPAS.fit.model.legacy	4
SPAS.print.model	6
SPAS.print.model.legacy	7

Index	8
--------------	----------

SPAS.fit.model	<i>Fit a Stratified-Petersen (SP) model using TMB.</i>
----------------	--

Description

This function fits a Stratified-Petersen (Plante, 1996) to data and specify which rows/columns of the data should be pooled. The number of rows after pooling should be \leq number of columns after pooling .

Usage

```
SPAS.fit.model(
  model.id = "Stratified Petersen Estimator",
  rawdata,
  row.pool.in,
  col.pool.in,
  row.physical.pool = TRUE,
  theta.pool = FALSE,
  CJSpool = FALSE,
  optMethod = c("nlminb"),
  optMethod.control = list(maxit = 50000),
  svd.cutoff = 1e-04,
  chisq.cutoff = 0.1
)
```

Arguments

model.id	Character string identifying the name of the model including any pooling..
rawdata	An $(s+1) \times (t+1)$ of the raw data BEFORE pooling. The $s \times t$ upper left matrix is the number of animals released in row stratum i and recovered in column stratum j . Row $s+1$ contains the total number of UNMARKED animals recovered in column stratum j . Column $t+1$ contains the number of animals marked in each row stratum but not recovered in any column stratum. The rawdata[$s+1$, $t+1$] is

not used and can be set to 0 or NA. The sum of the entries in each of the first s rows is then the number of animals marked in each row stratum. The sum of the entries in each of the first t columns is then the number of animals captured (marked and unmarked) in each column stratum. The row/column names of the matrix may be set to identify the entries in the output.

`row.pool.in`, `col.pool.in`

Vectors (character/numeric) of length s and t respectively. These identify the rows/columns to be pooled before the analysis is done. The vectors consists of entries where pooling takes place if the entries are the same. For example, if $s=4$, then `row.pool.in = c(1,2,3,4)` implies no pooling because all entries are distinct; `row.pool.in=c("a","a","b","b")` implies that the first two rows will be pooled and the last two rows will be pooled. It is not necessary that row/columns be continuous to be pooled, but this is seldom sensible. A careful choice of pooling labels helps to remember what is done, e.g. `row.pool.in=c("123","123","123","4")` indicates that the first 3 rows are pooled and the 4th row is not pooled. Character entries ensure that the resulting matrix is sorted properly (e.g. if `row.pool.in=c(123,123,123,4)`, then the same pooling is done, but the matrix rows are sorted rather strangely.

`row.physical.pool`

Should physical pooling be done (default) or should logical pooling be done. For example, if there are 3 rows in the data matrix and `row.pool.in=c(1,1,3)`, then in physical pooling, the entries in rows 1 and 2 are physically added together to create 2 rows in the data matrix before fitting. Because the data has changed, you cannot compare physical pooling using AIC. In logical pooling, the data matrix is unchanged, but now parameters $p_1=p_2$ but the movement parameters for the rest of the matrix are not forced equal.

`theta.pool`, `CJSpool`

NOT YET IMPLEMENTED. DO NOT CHANGE.

`optMethod`

What optimization method is used. Defaults is the `nlminb()` function..

`optMethod.control`

Control parameters for optimization method. See the documentation on the different optimization methods for details.

`svd.cutoff`

When finding the variance-covariance matrix, a singular value decomposition is used. This identifies the smallest singular value to retain.

`chisq.cutoff`

When finding a goodness of fit statistic using $(\text{obs}-\text{exp})^2/\text{exp}$, all cell whose $\text{Exp} < \text{gof.cutoff}$ are ignored to try and remove structural zero cells.

Value

A list with many entries. Refer to the vignettes for more details.

Examples

```
conne.data.csv <- textConnection("
9 , 21 , 0 , 0 , 0 , 0 , 171
0 , 101 , 22 , 1 , 0 , 0 , 763
0 , 0 , 128 , 49 , 0 , 0 , 934
0 , 0 , 0 , 48 , 12 , 0 , 434
0 , 0 , 0 , 0 , 7 , 0 , 49
```

```

0 , 0 , 0 , 0 , 0 , 0 , 4
351, 2736 , 3847 , 1818 , 543 , 191 , 0")
conne.data <- as.matrix(read.csv(conne.data.csv, header=FALSE))
close(conne.data.csv)

mod1 <- SPAS.fit.model(conne.data, model.id="Pooling rows 1/2, 5/6; pooling columns 5/6",
  row.pool.in=c("12", "12", "3", "4", "56", "56"),
  col.pool.in=c(1,2,3,4,56,56))

```

SPAS.fit.model.legacy *Fit a Stratified-Petersen (SP) model using the legacy method without TMB. The preferred method is to use the SPAS.fit.model() function that used TMB.*

Description

This function fits a Stratified-Petersen (Plante, 1996) to data and specify which rows/columns of the data should be pooled. The number of rows after pooling should be \leq number of columns after pooling .

Usage

```

SPAS.fit.model.legacy(
  model.id = "Stratified Petersen Estimator",
  rawdata,
  row.pool.in,
  col.pool.in,
  row.physical.pool = TRUE,
  theta.pool = FALSE,
  CJSpool = FALSE,
  sd.noise.init.est = 0,
  optMethod = c("BBoptim", "optim"),
  optMethod.control = list(maxit = 50000, ftol = 1e-09, gtol = 1e-05),
  svd.cutoff = 1e-04,
  chisq.cutoff = 0.1
)

```

Arguments

model.id	Character string identifying the name of the model including any pooling..
rawdata	An $(s+1) \times (t+1)$ of the raw data BEFORE pooling. The $s \times t$ upper left matrix is the number of animals released in row stratum i and recovered in column stratum j . Row $s+1$ contains the total number of UNMARKED animals recovered in column stratum j . Column $t+1$ contains the number of animals marked in each row stratum but not recovered in any column stratum. The <code>rawdata[s+1, t+1]</code> is not used and can be set to 0 or NA. The sum of the entries in each of the first s rows is then the number of animals marked in each row stratum. The sum of

the entries in each of the first t columns is then the number of animals captured (marked and unmarked) in each column stratum. The row/column names of the matrix may be set to identify the entries in the output.

`row.pool.in`, `col.pool.in`

Vectors (character/numeric) of length s and t respectively. These identify the rows/columns to be pooled before the analysis is done. The vectors consists of entries where pooling takes place if the entries are the same. For example, if $s=4$, then `row.pool.in = c(1,2,3,4)` implies no pooling because all entries are distinct; `row.pool.in=c("a","a","b","b")` implies that the first two rows will be pooled and the last two rows will be pooled. It is not necessary that row/columns be continuous to be pooled, but this is seldom sensible. A careful choice of pooling labels helps to remember what as done, e.g. `row.pool.in=c("123","123","123","4")` indicates that the first 3 rows are pooled and the 4th row is not pooled. Character entries ensure that the resulting matrix is sorted properly (e.g. if `row.pool.in=c(123,123,123,4)`, then the same pooling is done, but the matrix rows are sorted rather strangely.

`row.physical.pool`

Should physical pooling be done (default) or should logical pooling be done. For example, if there are 3 rows in the data matrix and `row.pool.in=c(1,1,3)`, then in physical pooling, the entries in rows 1 and 2 are physically added together to create 2 rows in the data matrix before fitting. Because the data has changed, you cannot compare physical pooling using AIC. In logical pooling, the data matrix is unchanged, but now parameters $p_1=p_2$ but the movement parameters for the rest of the matrix are not forced equal.

`theta.pool`, `CJSpool`

NOT YET IMPLEMENTED. DO NOT CHANGE.

`sd.noise.init.est`

How much random noise should be added to the initial (least squares) estimates. Normally only used with severe convergence problems.

`optMethod`

What optimization method is used. Defaults is the `BBoptim` function from the `BB` package.

`optMethod.control`

Control parameters for optimization method. See `spg()` function in `BB` package or `optim()` function for details. For `BBoptim`, a suggest control parameter for debugging is `optMethod.control=list(M=20, trace=TRUE, maxit = 50000, ftol=10^-5)`.

`svd.cutoff`

When finding the variance-covariance matrix, a singular value decomposition is used. This identifies the smallest singular value to retain.

`chisq.cutoff`

When finding a goodness of fit statistic using $(\text{obs}-\text{exp})^2/\text{exp}$, all cell whose $\text{Exp} < \text{gof.cutoff}$ are ignored to try and remove structural zero cells.

Value

A list with many entries. Refer to the vignettes for more details.

Examples

```
conne.data.csv <- textConnection("
```

```

9 , 21 , 0 , 0 , 0 , 0 , 171
0 , 101 , 22 , 1 , 0 , 0 , 763
0 , 0 , 128 , 49 , 0 , 0 , 934
0 , 0 , 0 , 48 , 12 , 0 , 434
0 , 0 , 0 , 0 , 7 , 0 , 49
0 , 0 , 0 , 0 , 0 , 0 , 4
351, 2736 , 3847 , 1818 , 543 , 191 , 0")
conne.data <- as.matrix(read.csv(conne.data.csv, header=FALSE))
close(conne.data.csv)

mod1 <- SPAS.fit.model.legacy(conne.data, model.id="Pooling rows 1/2, 5/6; pooling columns 5/6",
row.pool.in=c("12","12","3","4","56","56"),
col.pool.in=c(1,2,3,4,56,56),
optMethod.control=list(ftol=.0001))

```

SPAS.print.model	<i>Print the results from a fit of a Stratified-Petersen (SP) model when using the TMB optimizer</i>
------------------	--

Description

This function makes a report of the results of the model fitting .

Usage

```
SPAS.print.model(x)
```

Arguments

x A result from the model fitting. See SPAS.fit.model.

Value

A report to the console. Refer to the vignettes.

Examples

```

conne.data.csv <- textConnection("
9 , 21 , 0 , 0 , 0 , 0 , 171
0 , 101 , 22 , 1 , 0 , 0 , 763
0 , 0 , 128 , 49 , 0 , 0 , 934
0 , 0 , 0 , 48 , 12 , 0 , 434
0 , 0 , 0 , 0 , 7 , 0 , 49
0 , 0 , 0 , 0 , 0 , 0 , 4
351, 2736 , 3847 , 1818 , 543 , 191 , 0")
conne.data <- as.matrix(read.csv(conne.data.csv, header=FALSE))
close(conne.data.csv)

mod1 <- SPAS.fit.model(conne.data, model.id="Pooling rows 1/2, 5/6; pooling columns 5/6",

```

```
row.pool.in=c("12","12","3","4","56","56"),
col.pool.in=c(1,2,3,4,56,56))
```

```
SPAS.print.model(mod1)
```

```
SPAS.print.model.legacy
```

Print the results from a fit of a Stratified-Petersen (SP) model from the legacy (i.e. pre using TMB) code

Description

This function makes a report of the results of the model fitting .

Usage

```
SPAS.print.model.legacy(x)
```

Arguments

x A result from the model fitting. See SPAS.fit.model.

Value

A report to the console. Refer to the vignettes.

Examples

```
conne.data.csv <- textConnection("
9 , 21 , 0 , 0 , 0 , 0 , 171
0 , 101 , 22 , 1 , 0 , 0 , 763
0 , 0 , 128 , 49 , 0 , 0 , 934
0 , 0 , 0 , 48 , 12 , 0 , 434
0 , 0 , 0 , 0 , 7 , 0 , 49
0 , 0 , 0 , 0 , 0 , 0 , 4
351, 2736 , 3847 , 1818 , 543 , 191 , 0")
conne.data <- as.matrix(read.csv(conne.data.csv, header=FALSE))
close(conne.data.csv)

mod1 <- SPAS.fit.model.legacy(conne.data, model.id="Pooling rows 1/2, 5/6; pooling columns 5/6",
row.pool.in=c("12","12","3","4","56","56"),
col.pool.in=c(1,2,3,4,56,56),
optMethod.control=list(ftol=.0001))

SPAS.print.model.legacy(mod1)
```

Index

SPAS.fit.model, [2](#)
SPAS.fit.model.legacy, [4](#)
SPAS.print.model, [6](#)
SPAS.print.model.legacy, [7](#)